title: Introduction

MEV-Share is a new protocol for sending and searching on Ethereum transactions. In this guide, we'll write our own bot to see first-hand how MEV-Share works, and how to find profitable opportunities.

This guide assumes you have some programming experience, but we've tried to make it as beginner-friendly as possible! No prior searching experience will be required.

We'll provide full examples with code so you can follow along and run the bot yourself!

Limit Orders

In this guide, we'll be making a version of a limit order bot. Limit orders are a common feature on exchanges which let you fill an order when the price of a trading pair (e.g. ETH/DAI) reaches a target that you specify. For example, if I wanted to buy DAI when the price reaches 1800 DAI/ETH, I could place a limit order for to buy 1800 DAI for 1 ETH, and the trade would automatically execute when the price reached 1800. If the price was over 1800, then we'd want to fill our order at the higher price — since we're buying DAI, we want more DAI out for a fixed amount of ETH in.

Limit orders are useful if you are sensitive to price but not time. For example, I might have a lot of ETH and I want to make sure I get a good price for it. But it's okay if it takes week to complete the trades.

The bot we're building works like a traditional limit order — we'll buy when the price reaches our target. But in this case, we'll have an additional edge: private orderflow (and some clever code).

MEV-Share Bot

We'll use the MEV-Share event stream to watch for pending transactions (private orderflow) that change the price of our trading pair. Then we'll backrun each of those transactions with our ideal trade. When a transaction sufficiently shifts the price in our favor, our backrun will be first in line to buy the tokens at a discounted rate.

Our backrun transaction will specify an exact price at which the order can be filled, otherwise the transaction reverts. Because we're sending to Flashbots, reverted transactions won't land on chain and we won't pay any fees for failed attempts.

In short, we'll attempt to backrun all trades for the assets we care about — but only land the backruns that execute at a desirable price.

> This guide is based on the simple-limit-order-bot repo. If you want to get straight to the code, the repo contains a fully-operational bot that only requires a `.env` file for you to run.

Glossary

Remember the following terminology (because we use it a lot!):

- **MEV**: Maximal Extractable Value.
- **bundle**: an array of transactions that execute in order andatomically.
- **backrun**: a transaction sent immediately after another transaction.
  - example: a “backrun bundle” is an array with two or more transactions; the first transaction (presumably chosen from a public pool of pending transactions) creates an MEV opportunity, which the following transactions try to capture.
- **orderflow**: umbrella term for transactions or bundles.
- **searcher**: a bot operator (or bot) that attempts to extract MEV.
- **on-chain**: a transaction that lands “on chain” (or on-chain, or onchain) is permanently included on the blockchain.
- **WETH**: “Wrapped ETH” — ERC-20 version of ETH, used by Uniswap in trading pairs
- **MEV-Share**: “a protocol that lets users selectively share orderflow and information with searchers”.

:::info Running this bot requires that you have an Ethereum account with some ETH and WETH.

Our example uses 1/10000000000 of an ETH — you may want more. If you don't have any WETH, you can wrap ETH into WETH on Uniswap:

::: info